1-1-2014

A critical systematic review of service concretization based on bio-inspired approaches

Lijuan Wang
University of Wollongong, lw840@uowmail.edu.au

Jun Shen
University of Wollongong, jshen@uow.edu.au

A critical systematic review of service concretization based on bio-inspired approaches

Abstract

In service-oriented computing, Web service selection is an important part of Web service composition. QoS attributes are used to distinguish Web services with same functionality. QoS-aware Web service selection and composition have attracted much attention in the past years. The Web service composition is achieved by solving the Web service concretization problem. The literature has presented two Web service concretization approaches, namely, local optimization approach and global optimization approach. There are three types of algorithmic methods in the global optimization approaches: optimal methods, sub-optimal methods, and soft constraints-based methods. The bio-inspired algorithms are sub-optimal methods. The existing studies using bio-inspired algorithms apply different strategies to solve Web service concretization problems. This paper will firstly present a hierarchical taxonomy of Web service concretization approaches. Then we present a systematic review on the current research of Web service concretization based on bio-inspired algorithms. We will describe the applications of ant colony optimization algorithm, genetic algorithm, and particle swarm optimization algorithm to Web service concretization. In addition, this paper will compare ant colony optimization algorithm and genetic algorithm in details. By analyzing and comparing different applications of bio-inspired algorithms-Web service concretization, this paper also discusses the known outcomes and areas for potential future research.

Keywords

systematic, approaches, critical, inspired, review, bio, concretization, service

Disciplines

Engineering | Science and Technology Studies

Publication Details

Wang, L. & Shen, J. (2014). A critical systematic review of service concretization based on bio-inspired approaches. 1-21.

A Critical Systematic Review of Service Concretization Based on Bio-inspired Approaches

Lijuan Wang, Jun Shen

Abstract

In service-oriented computing, Web service selection is an important part of Web service composition. QoS attributes are used to distinguish Web services with same functionality. QoS-aware Web service selection and composition have attracted much attention in the past years. The Web service composition is achieved by solving the Web service concretization problem. The literature has presented two Web service concretization approaches, namely, local optimization approach and global optimization approach. There are three types of algorithmic methods in the global optimization approaches: optimal methods, sub-optimal methods, and soft constraints-based methods. The bio-inspired algorithms are sub-optimal methods. The existing studies using bio-inspired algorithms apply different strategies to solve Web service concretization problems. This paper will firstly present a hierarchical taxonomy of Web service concretization approaches. Then we present a systematic review on the current research of Web service concretization based on bio-inspired algorithms. We will describe the applications of ant colony optimization algorithm, genetic algorithm, and particle swarm optimization algorithm to Web service concretization. In addition, this paper will compare ant colony optimization algorithm and genetic algorithm in details. By analyzing and comparing different applications of bio-inspired algorithms-Web service concretization, this paper also discusses the known outcomes and areas for potential future research.

Index Terms

ant colony optimization, genetic algorithm, particle swarm optimization, Web service concretization, Web service selection, quality of service

1 Introduction

As a major software framework for distributed applications, service-oriented architecture (SOA) [21] uses standard protocols to integrate existing component Web services into complex business processes and applications, which are referred as Web service composition. These component Web services are developed independently by different service providers, so some services may have same functionality but differ in quality of service (QoS) as well as other non-functional properties. QoS-aware Web service selection and composition have attracted much attention in the past years. Web service selection is an important part of Web service composition. Given a request of composite service, which involves a set of abstract tasks and dependency relationships among them, while there are a list of service candidate sets, which include many service candidates for each abstract task. Web service selection refers to finding one service candidate to implement each abstract task according to users' requirements. The final goal of the composite service construction is achieved by solving the well-known service concretization problem. So far, many service concretization approaches have been designed. A hierarchical taxonomy of service concretization approaches will be given in this paper. The bio-inspired algorithms are one of the main approaches.

One of our earlier studies [61] presented a survey on bio-inspired algorithms for Web service composition, and it also proposed a bio-inspired cost minimization mechanism for data-intensive service provision. However, it did not describe the applications of the presented algorithms to Web service composition specifically. There are also a few other overviews on service composition and bio-inspired optimization algorithms. The study [49] presented an overview of automatic Web service composition from both the workflow and AI planning research communities. The authors of [20] discussed the urgent need for service composition, the

- *L. Wang, and J. Shen are with the Soci-Technical Information Systems Center, School of Information Systems and Technology, University of Wollongong, NSW, Australia.*
  E-mail: lw840@uowmail.edu.au, jshen@uow.edu.au

required technologies to perform service composition, as well as several different composition strategies based on existing composition platforms and frameworks. The study [56] presented a technical survey on automated service composition and discussed whole automatic composition from service classification, service combination, service selection, service description, and service matchmaking.

However, to the best of our knowledge, no overview on service concretization problem based on bio-inspired algorithms has been published yet. This paper investigates the existing studies using bio-inspired algorithms to deal with QoS, optimality, and dynamicity in Web service concretization. We will describe the applications of ant colony optimization (ACO) algorithm, genetic algorithm (GA), and particle swarm optimization (PSO) algorithm to Web service concretization. In addition, this paper will compare ACO algorithm with GA. The main contributions of this paper are three folds: 1) A hierarchical taxonomy of service concretization approaches is given. 2) A systematic survey and analysis of Web service concretization based on bio-inspired algorithms are presented. 3) Comparisons between ACO-based and GA-based Web service selection are conducted. This paper gives an overview for people interested in bio-inspired service concretization. Moreover, through this paper we can understand QoS-based Web service selection and composition more comprehensively.

The paper is organized as follows: Section 2 presents the background. Section 3 introduces a hierarchical taxonomy of service concretization approaches. Section 4 describes our systematic review protocol. A number of studies based on bio-inspired algorithms to solve Web service selection and composition are given in Section 5. It shows how bio-inspired algorithms have been successfully applied to Web service concretization problems. Then comparisons between ACO and GA are given in Section 6. Finally, Section 7 concludes this paper.

2 Background

In a Web service environment, abstract services are the functional descriptions of services, and concrete services represent the existing services, which are available for potential invocation of their functionalities and capabilities. When the functions of several concrete services are consistent with the functional description of an abstract service, these concrete services are the service candidates of the abstract service and QoS attributes are used to distinguish them.

2.1 Classification of QoS Attributes

QoS attributes for services refer to various non-functional properties. QoS attributes can be divided into two groups: quantitative attributes and qualitative attributes [41]. The former attributes such as response time and availability are quantitatively measured using metrics, whereas the latter attributes such as security cannot be measured. For the sake of simplicity and without loss of generality, in this paper only quantitative QoS attributes are considered, since qualitative attributes can be represented as quantitative attributes by using specific functions. For example, the flexibility attribute has the value type {inflexible, flexible, very flexible} which can be mapped to the set {0, 1, 2}. The quantitative QoS attributes are in turn divided into two classes: negative attributes and positive attributes. The first class of attributes such as response time and cost have a negative effect on QoS, hence they should be optimally minimized because the higher the value, the lower the quality. On the contrary, positive QoS attributes such as availability and reliability should be maximized.

2.2 Evaluation Function for Concrete Service

For an abstract service AS_i, there is a service candidate set of $cs_i = \{cs_{i,1}, cs_{i,2}, \ldots, cs_{i,m}\}$ that can be used to realize AS_i. Each concrete service $cs_{i,j}$ is associated with a QoS vector $q_{cs_{i,j}} = [q^1_{cs_{i,j}}, q^2_{cs_{i,j}}, \ldots, q^r_{cs_{i,j}}]$ with r QoS parameters ($i \in \{1, 2, \ldots, n\}$, $j \in \{1, 2, \ldots, m\}$). In order to evaluate the multidimensional quality of concrete service $cs_{i,j}$, an evaluation function is used. The function maps the quality vector $q_{cs_{i,j}}$ into a single real value to enable selection of service candidates. In this paper, a multiple attribute decision-making approach for the evaluation function is used, that is, the simple additive weighting (SAW) technique [69]. There are two phases in applying SAW:

1) Scaling phase. As already mentioned, QoS attributes can be either negative or positive. For negative QoS attributes, values are scaled according to (1). For positive QoS attributes, values are scaled according to (2).

$$V^k_{cs_{i,j}} = \begin{cases} \dfrac{Q^{max}_{k,i} - q^k_{cs_{i,j}}}{Q^{max}_{k,i} - Q^{min}_{k,i}} & \text{if } Q^{max}_{k,i} - Q^{min}_{k,i} \neq 0 \\ 1 & \text{if } Q^{max}_{k,i} - Q^{min}_{k,i} = 0 \end{cases} \quad (1)$$

$$V_{cs_{i,j}}^{k} = \begin{cases} \dfrac{q_{cs_{i,j}}^{k} - Q_{k,i}^{min}}{Q_{k,i}^{max} - Q_{k,i}^{min}} & \text{if } Q_{k,i}^{max} - Q_{k,i}^{min} \neq 0 \\ 1 & \text{if } Q_{k,i}^{max} - Q_{k,i}^{min} = 0 \end{cases} \quad (2)$$

In (1) and (2), $Q_{k,i}^{max}$ and $Q_{k,i}^{min}$ ($k \in \{1, 2, \ldots, r\}$) are used to represent the maximal value and the minimal value of the k-th QoS attribute of all concrete services in candidate set cs_i, which are given by (3).

$$\begin{aligned} Q_{k,i}^{min} &= \min_{\forall cs_{i,j} \in cs_i} q_{cs_{i,j}}^{k} \\ Q_{k,i}^{max} &= \max_{\forall cs_{i,j} \in cs_i} q_{cs_{i,j}}^{k} \end{aligned} \quad (3)$$

2) Weighting phase. The overall score of $cs_{i,j}$ is computed according to (4).

$$score_{cs_{i,j}} = \sum_{k=1}^{r} (V_{cs_{i,j}}^{k} * W_k) \quad (4)$$

where $W_k \in [0, 1]$ and $\sum_{k=1}^{r} W_k = 1$. W_k represents the weight of k-th quality criteria with value normally provided by the users based on their own preferences.

2.3 Web Service Composition Structures

A set of basic workflow patterns commonly used by composition approaches [70], [72], which cover most of the structures specified by composition languages (such as YAWL [58] and BPEL) are considered in this paper. They are: sequence, parallel split (AND-split), synchronization (AND-join), exclusive choice (XOR-split), simple merge (XOR-join), and loop.

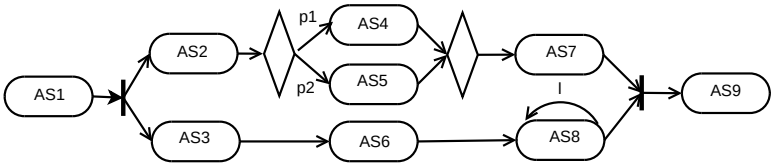

Fig. 1: An instance of workflow for service composition

A simplified workflow is depicted in Fig. 1, where the irregular circles represent an abstract service (denoted by AS_i). Using the probability (p_1 and p_2) of execution of XOR branches and the expected maximum iteration numbers (l) of the loop pattern, the workflow of a composite process can be transformed to a directed acyclic graph (DAG) through the use of loop pealing [8].

After loop pealing, from the transformed DAG, a set of execution paths that identify all possible execution scenarios of the composite service can be derived. An execution path is a sequence of abstract services $\{AS_1, AS_2, \ldots, AS_n\}$ where AS_1 and AS_n are the initial and final abstract service, respectively, and no abstract service AS_{i_1} and AS_{i_2} belong to alternative branches. Every execution path has an associated probability of execution that can be evaluated as the product of the probability of execution of the branch conditions included in the execution path. There are two execution paths, called ep_1 and ep_2 in Fig. 1.

$ep_1 : \{AS_1, AS_2, AS_3, AS_4, AS_6, AS_7, l * AS_8, AS_9\}$, with a probability of p1;
$ep_2 : \{AS_1, AS_2, AS_3, AS_5, AS_6, AS_7, l * AS_8, AS_9\}$, with a probability of p2.

For each abstract service AS_i of an execution path, a concrete service is selected from the service candidate set associated with abstract service AS_i, then a concrete execution plan is derived.

2.4 Computing the QoS of Composite Services

The performance of composite services is directly influenced by the quality of component services. The evaluation of the QoS of possible composite services depends on the workflow patterns and also QoS aggregation formulas associated with each pattern.

The overall QoS of a composite service can be computed by applying the functions described in Table 1, which shows an aggregation formula for each workflow pattern and QoS attribute. For a sequence construct of services s_i and $i \in \{1, \ldots, ns\}$, the RT and C functions are additive while Av and Re are multiplicative. For a XOR (also called conditional) construct of services s_i and $i \in \{1, \ldots, nc\}$, the QoS attributes are always

TABLE 1: QoS aggregation functions for different workflow patterns [15]

QoS Attributes	Workflow Patterns			
	Sequence	XOR	AND	Loop
Response time (RT)	$\sum_{i=1}^{ns} RT(s_i)$	$\sum_{i=1}^{nc} pc_i * RT(s_i)$	$Max\{RT(s_i)_{i\in\{1,\ldots,np\}}$	$l * RT(s)$
Cost (C)	$\sum_{i=1}^{ns} C(s_i)$	$\sum_{i=1}^{nc} pc_i * C(s_i)$	$\sum_{i=1}^{np} C(s_i)$	$l * C(s)$
Availability (Av)	$\prod_{i=1}^{ns} Av(s_i)$	$\sum_{i=1}^{nc} pc_i * Av(s_i)$	$\prod_{i=1}^{np} Av(s_i)$	$[Av(s)]^l$
Reliability (Re)	$\prod_{i=1}^{ns} Re(s_i)$	$\sum_{i=1}^{nc} pc_i * Re(s_i)$	$\prod_{i=1}^{np} Re(s_i)$	$[Re(s)]^l$

evaluated as a sum of the attribute value of each service, times the probability of the case (pc_i, and $\sum_{i=1}^{nc} pc_i = 1$) to which it belongs. The aggregation functions for the AND (also called parallel) construct, are essentially the same as those for the sequence construct, except for the RT attribute where this is the maximum time of the parallel services s_i and $i \in \{1, \ldots, np\}$. Finally, a loop construct with l iterations of service s is equivalent to a sequence construct of l copies of s. Table 1 only contains four QoS attributes. It should be noted that other attributes and their aggregation functions can also be specified similarly.

2.5 Service Concretization Problem

According to service concretization problem, the best service available at runtime has to be selected among all candidate ones for each abstract service, taking into consideration the global and local QoS constraints given by the user. The main decision variables of service concretization problem are $\sum_{j=1}^{m} x_{i,j} = 1$ (m is the number of service candidates for abstract service AS_i), where $x_{i,j}$ is set to 1 if abstract service AS_i is implemented by concrete service $cs_{i,j}$ and 0 otherwise.

The goal of service concretization problem is to maximize the aggregated quality by considering all possible execution paths of the abstract execution plan and their corresponding probability of execution. Each execution path ep_z is associated with a QoS vector $q_{ep_z} = [q^1_{ep_z}, q^2_{ep_z}, \ldots, q^r_{ep_z}]$ and a probability of execution p_{ep_z}. The global QoS constraints define requirements regarding the aggregated QoS values of the requested composite service, which are expressed in terms of a vector $Q_c = [Q^1_c, Q^2_c, \ldots, Q^u_c]$, $1 \leq u \leq r$. Then the service concretization problem can be stated as follows.

$$\max \sum_{z=1}^{Z} p_{ep_z} * score_{ep_z} \tag{5}$$

subject to:

$$q^k_{ep_z} \leq Q^k_c, \quad \forall Q^k_c \in Q_c, \text{k is negative attributes;}$$
$$q^k_{ep_z} \geq Q^k_c, \quad \forall Q^k_c \in Q_c, \text{k is positive attributes.}$$

where $score_{ep_z}$ is the overall score of execution path ep_z, and Z is the total number of execution paths.

The score of an execution path is also computed according to SAW technique, which is similar to (1) - (4). When scaling the aggregated values of QoS attributes for an execution path, the minimum and maximum value of the k-th QoS attribute given by (3) should replaced by (6).

$$Q^{MIN}_k = F(k)^n_{i=1} Q^{min}_{k,i}$$
$$Q^{MAX}_k = F(k)^n_{i=1} Q^{max}_{k,i} \tag{6}$$

where the function $F(k)$ denotes the aggregation function of the k-th QoS attribute, which can refer to Table 1.

3 Service Concretization Approaches

The literature has presented two service concretization approaches, namely, local optimization approach and global optimization approach. Fig. 2 gives a hierarchical taxonomy of service concretization approaches.

3.1 Local Optimization Approach

In local optimization approaches, service selection is done for each abstract service independently [10], [35], [45], [72]. The best candidate is selected for each abstract service. This approach is very efficient in terms of computation time, as the time complexity of local optimization approach is $O(m)$ by using service broker, peer-to-peer or agent-based architecture to perform selection in parallel, where m is the number of service candidates for each abstract service. Even if the approach is useful in decentralized and dynamic environments,

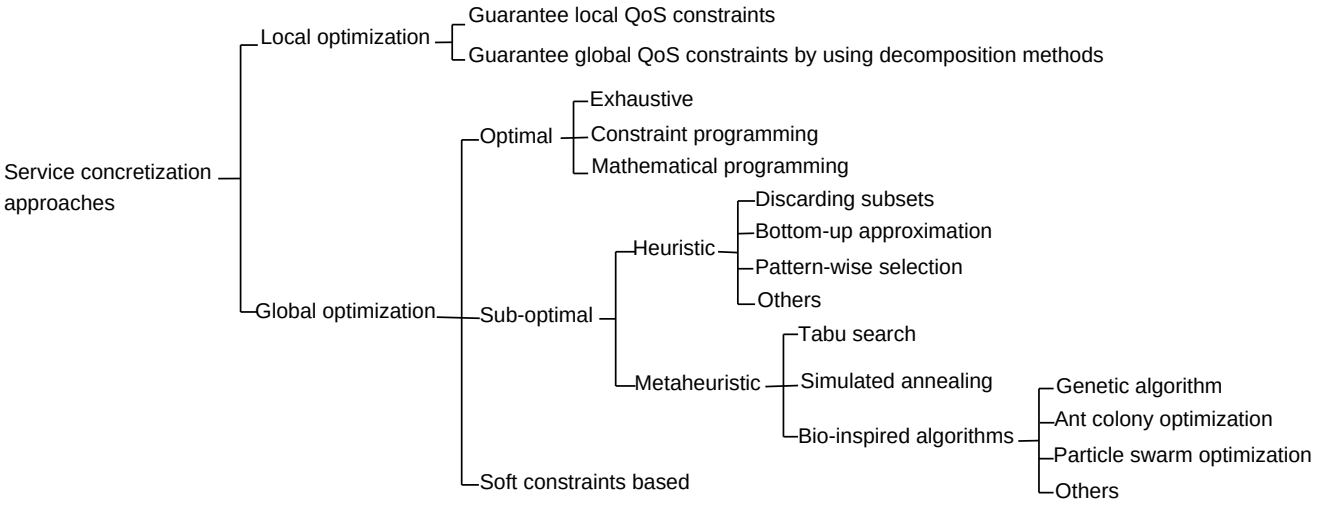

Fig. 2: A hierarchical taxonomy of service concretization approaches

local optimization approach is not suitable for QoS-based service selection with global QoS constraints, since it can only guarantee local QoS constraints and cannot satisfy the global QoS constraints and preferences.

From the viewpoint of computational time, the local optimization approach can be appropriate when the global QoS constraints are transformed into local constraints in order to overcome its drawback. Some decomposition methods have been proposed [5], [42], [55].

The authors of [5] used the concept of local quality levels, and modeled the decomposition problem as an optimization problem which was formulated as a mixed-integer programming model. The local quality levels were a set of discrete representative values extracted from the quality properties of all candidates. The local constraints were the selected quality levels. The study [42] modeled the decomposition problem as an optimization problem which was solved by a genetic algorithm. They used the concept of quality degree. The quality degrees were a set of discrete quality values from partitioning each quality property. The local constraints were the selected quality degrees. In [55], the mean and standard deviation of the values of QoS attributes were used to decompose global QoS constraints. The time complexity of the decomposition approach in [5] is $O(n*c*d)$, in [42] is $O(n*c*q)$, and in [55] is $O(n*c*m)$, where n is the number of abstract services, c is the number of global QoS constraints, d is the number of quality levels, q is the number of quality degrees, and m is the number of candidates in each service candidate set.

The main drawbacks of these decomposition methods are that the local selection relies on a greedy method, the decomposition deals with QoS parameters independently and does not take into account potential correlations and dependencies among them. Meanwhile, in the above decomposition methods there is no analysis for cases when the requirements of the global QoS constraints are over-constrained. This often leads to a very restrictive decomposition of the global QoS constraints that cannot be satisfied by any of the service candidates even for the otherwise feasible problems.

3.2 Global Optimization Approach

Global optimization approaches can solve QoS-aware service selection problem at both local and global service levels [7], [14], [30], [72]. When using global optimization approaches to solve the Web service selection problem, it has been modeled as a 0-1 knapsack problem [31], [34], [70], a constraint optimization problem [25], [57], [71], a multi-dimensional and/or multi-objective optimization problem [36], [39], [74], a weighted directed acyclic graph problem [47], [67], and a mathematical programming problem [1], [6], [7], [17], [27], [48], [72]. Accordingly, many algorithms have been proposed to solve these problems.

There are three types of algorithmic methods in the global optimization approaches: optimal method, sub-optimal method, and soft constraints-based method. Optimal methods, such as the exhaustive algorithm [22], constraint programming-based algorithm [28], [52], and mathematical programming-based algorithm (integer programming (IP) [1], [72], linear programming (LP) [17], or mixed integer programming (MIP) [6], [7], [48]) are designed to find optimal solutions. The authors of [52] explained that constraint programming-based algorithms were essentially very simple, sometimes they could find a solution faster than more complex mathematical programming-based algorithms. Sub-optimal methods, such as the heuristic-based algorithm (discarding subsets, bottom-up approximation and pattern-wise selection [31], H1_RELAX_IP [11], shortest

path heuristic [13], and other heuristics [41], [70]), and the meta-heuristic-based algorithm (tabu search [44], simulated annealing [23], genetic algorithm [14], [40], ant colony optimization [65], [74], particle swarm optimization [23], [75] and others [33]) are designed to find an optimal or a near optimal solution. Meta-heuristics provide both a general structure and strategy guidelines for developing a heuristic for solving service concretization problems. They provide an efficient way to move toward a very good solution. Soft constraints-based algorithms [25], [46], [68], [71] are adopted in order to allow constraint violations in less likely feasible compositions when the requirements of the global QoS constraints are over-constrained.

3.3 Existing Service Concretization Approaches

QoS-aware service composition problem is well-known as a NP-hard problem with no known polynomial-time algorithms to solve them [29]. Besides, the composition can be either static or dynamic. In static composition, all services and their implementations are available before the application is composed. No runtime reconfigurations are possible once the composite service application is launched. The static composition is not concerned with the complexity but the focus is more on the accuracy/optimality of the approach. In dynamic composition, physical nodes, services, and service implementations can be replaced or reconfigured, all of these lead QoS attributes of services to change. Thus the service composition has to be re-planned and the concerns more on the complexity.

The service concretization approaches will be judged with respect to their optimality, their computational efficiency, and their complexity. The optimality is the extent to which the approach produces the best solution. The computational efficiency is the time the approach takes to produce a solution. The complexity is how the approach deals with the re-optimization. By analyzing the existing service concretization approaches by the above three aspects, it can be seen the following results.

1) If there is no requirement specifying global constraints, then local optimization approach is preferable.
2) Optimal method such as mathematical programming-based algorithm, which find optimal solutions are often practically useful only for small scale problems. When the time requirements become excessive or strictly prohibitive, sub-optimal methods must be employed. Continuing efforts have demonstrated that sub-optimal methods for QoS-aware service composition usually require much less computation time than optimal methods and are consistently capable of achieving near optimal solutions [11], [13], [18], [22], [29], [31].
3) The optimal methods need more computation time, especially in dynamic environments. This is exacerbated in a situation where the QoS-aware composition has to be re-planned at runtime, since the computation time of the composition becomes crucial. This was confirmed by the existing studies [14], [22], [23], [70], [72].
4) IP, LP and MIP-based algorithms need linear aggregation functions for the QoS attributes. Some standard attributes, such as the availability or reliability are not linear initially. While linearization can be adopted [72], it is quite difficult in other cases, such as computing the response time for the parallel structures. An alternative would be the use of non-linear IP, however scalability problems would still arise [14]. When using constraint programming-based algorithms the constraints can be expressed without translating them into linear inequalities [52].
5) In the case of the sub-optimal methods (heuristic-based or meta-heuristic-based algorithms), there is a trade-off between computation cost and quality of the solution (how worse sub-optimal methods would be, when compared to an optimal method that always finds the optimal solution).
6) Soft constraints-based methods are proposed to predict run-time service level agreement (SLA) violations for composite services. They work around over-constrained QoS-aware service composition problem and offer a feasible solution. The choice of evaluation functions, defining penalties, and finding a relation between the assigned penalties and the violated guarantees are key factors in soft constraints-based methods.

3.4 Bio-inspired Algorithm-based Service Concretization

In this paper, we are interested in how bio-inspired algorithms are used to solve Web service concretization problems. Although numerous research works have addressed Web service selection and composition problem (including bio-inspired approaches), service discovery and composition mechanisms will face new challenges as the development of cloud computing, big data, and future Internet. We believe that applying biologically inspired approaches to services can make them adapt to dynamic service environment. For example, the authors of [9] had proved that it was beneficial to service management and discovery by adding biological mechanisms to services.

Biological systems present features such as autonomy, scalability, adaptability, and robustness. They are autonomous entities and often self-organized without a central controller. Typical life cycles and behaviors of biological organisms include: birth and death, migration, and replication (or division) of a group of organisms [9]. The biological environment provides a medium that allows biological organisms to interact and mobilize. For example, ants release chemical signals creating chemical gradients in the environment, letting other ants sense the chemical and follow the path to food discovered by other ants.

The author of [43] pointed out four key attributes of biological systems which can be applied to the design of biological inspired systems.

1) A large number of redundant components: a biological system is composed of massive numbers of redundant components. For example, an ant colony may contain millions of ants. A biological system is robust under perturbations or conditions of uncertainty because of the large number of redundant components.
2) Local interactions and collective behavior: an individual ant is not able to find the shortest path to food source, however, a group of ants can find the shortest path via interactions among individual ants through the placement of pheromones.
3) Stochastic or probabilistic nature: the probabilistic nature of biological systems helps them explore large search spaces. For example, mutation in genetic evolution randomly changes genetic information to enable individuals to adapt to the environment. Ants find the shortest path via the probability of choosing different paths.
4) Feedback-based control: biological systems frequently use positive and negative feedback control. For example, ants use positive feedback to deposit the pheromone and to recruit more ants toward a particular path.

In this paper, we will discuss these characteristics in three biological systems, namely, insect colonies, genetic evolution, and swarms. With bio-inspired algorithm-based Web service selection and composition, it is both important and interesting to know the extent of application of bio-inspired algorithms to QoS-based Web service concretization, which are not covered by previous studies. This effort allows us to provide an overview on the existing Web service concretization problem using bio-inspired algorithms.

4 Method of Systematic Review

This section describes our systematic review protocol, consisting of several steps as outlined in [32].

4.1 Research Questions

In order to find principles for applying bio-inspired algorithms to solve Web service concretization problems, we have the following research questions:

1) How are the three types of bio-inspired algorithms applied to solve Web service concretization problems? After knowing how each type of bio-inspired algorithm to solve Web service concretization problem, we have three additional research questions applicable in each type:
2) What are the main issues that need to be addressed if the three types of bio-inspired algorithms are to be applied?
3) What are the strategies proposed to address the issues raised?
4) What are the current challenges or limitations in the application of the three types of bio-inspired algorithms for Web service concretization problems?

The domain of this study is the Web service composition and selection based on ACO, GA, and PSO. Our research questions are not aimed at making a comparison among the three algorithms, however, we discuss the comparisons within the scope of each primary study to support our argumentation of obtained results, and we will present comparisons by experiments.

4.2 Specification for Literature Review Strategy

We use the following terms to search the literature:

1) Web service composition, Web service selection.
2) ant colony algorithm, ant colony optimization, ant-inspired, ant system, ant-based, genetic algorithm, particle swarm optimization, particle swarm algorithm.

We used five electronic databases, namely, ACM Digital Library, ScienceDirect, Google Scholar, IEEExplore, and EI Compendex as data sources and papers were selected for review if they had a key term (or synonym) in the title, abstract or keywords list. We selected 2005 as the starting year for the search since this year marked

TABLE 2: Distribution of papers before and after duplication removal among different publication sources

Source	Count
ACM Digital Library	10(10)
ScienceDirect	11(11)
Google Scholar	55(54)
IEEExplore	154(120)
EI Compendex	224(83)
Total	454(278)

the first publication of the application of genetic algorithm to Web service composition, and the ending year is 2012. The search resulted in a total of 454 papers. After eliminating duplicates found by more than one electronic database, we were left with 278 papers. Table 2 shows the distribution of papers before and after removal of duplications among different sources.

The following exclusion criteria is applicable in this review to exclude studies that:

1) Do not relate to Web service selection and composition.
2) Do not report application of bio-inspired algorithms.
3) Do not report experiments and simulations.
4) Are related to semantic Web service composition and selection.

The application of detailed exclusion criteria resulted in 120 remaining references, which were further filtered out by reading full-text. A final figure of 22 primary studies was reached after excluding similar studies that were published in different venues. Relevant information describing the distribution of primary studies within the four QoS attributes is shown in Table 3. The four QoS attributes which are the most commonly used are: response time, cost, reliability and availability, and according to [30], they represent a selection of the most relevant characteristics in the field of services. Response time, run time, and execution time might have the same or different definitions. Table 3 places these attributes into the single time column. In addition, the studies [23], [24], [38] did not provide details about whether the four QoS attributes were considered or not.

TABLE 3: Distribution of primary studies

Bio-inspired algorithm	Articles	Year	Cost	Time	Availability	Reliability
ACO	[76]	2007	∨	∨	∨	
	[66]	2008	∨	∨	∨	∨
	[60]	2010	∨	∨		∨
	[74]	2010	∨	∨	∨	∨
	[65]	2011	∨	∨	∨	∨
GA	[14]	2005	∨	∨	∨	∨
	[16]	2005	∨			
	[73]	2006	∨	∨	∨	∨
	[26]	2007	∨	∨	∨	∨
	[2]	2008	∨	∨	∨	∨
	[44]	2008	∨	∨		∨
	[40]	2008	∨	∨	∨	∨
	[37]	2009	∨	∨	∨	∨
	[57]	2010	∨	∨	∨	∨
	[59]	2012		∨	∨	∨
PSO	[24]	2009				
	[50]	2010	∨	∨		
	[64]	2012	∨	∨	∨	
	[75]	2012	∨	∨	∨	∨
Combination of GA and PSO	[38]	2007				
Combination of ACO and GA	[67]	2010	∨	∨		
Combination of SPSO and SA	[23]	2011				

5 Results and Synthesis of Findings

5.1 Ant Colony Optimization Algorithm

The ant colony algorithm modeling the behavior of real ants is widely used for combinatorial optimization problem. Ant colony algorithms are developed as heuristic methods to identify efficient paths through a graph and have been applied to identify optimal solutions for service composition problems. The features of ant colony algorithms include positive feedback and local heuristics.

When ant colony algorithms are used to solve Web service concretization problem, the problem is modeled as a weighted DAG with a start point S and a target point T. Then the start point S is set as the ants' nest and the target point T is set as the food source, and the QoS constraints are regarded as the weights of the edges. Thus, the problem is transformed from selecting optimal schema for composite service into selection of the optimal path in the weighted DAG. Fig. 3 presents a service composition graph.

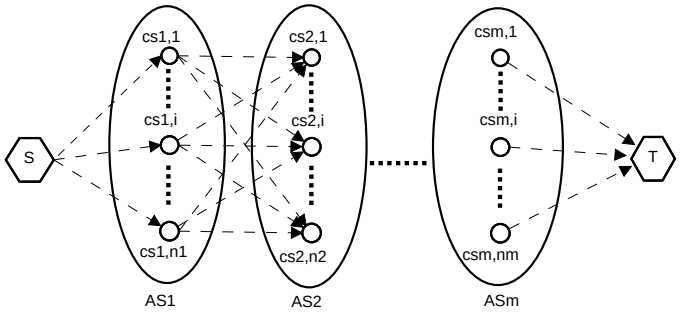

Fig. 3: A service composition graph

The details of the simple ACO algorithm which adapts the real ants' behavior to the solution of optimal path problems on weighted DAG are presented in the following. Given a DAG G=<V, E, QoS>, two nodes $i, j \in V$ are neighbors if there exists an arc $(i, j) \in E$. τ_{ij} is the pheromone density on the edge (i, j). At the beginning of the search process, a constant amount of pheromone is assigned to all the arcs, namely, $\tau_{ij} = C$ (C is a constant, $\forall (i, j) \in E$). In the process of searching, when located at a node i, each ant k uses the pheromone density τ_{ij} and the heuristic information η_{ij} on the path to compute the probability of choosing j as next node:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, & \text{if } j \in N_i^k \ ; \\ 0, & \text{otherwise .} \end{cases} \tag{7}$$

where p_{ij}^k represents the probability from point i to point j of ant k, N_i^k is the neighborhood of ant k when it is in node i, α is a parameter to control the influence of τ_{ij}, β is a parameter to control the influence of η_{ij}. The neighborhood of a node i contains all the nodes directly connected to node i in the graph G, except for the predecessor of node i.

Each ant changes the pheromone value τ_{ij} according to (8).

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau^k, \forall (i, j) \in E. \tag{8}$$

where ρ is the pheromone evaporation coefficient, $\Delta\tau^k$ is the amount of pheromone deposited by ant k. The choice of $\Delta\tau^k$ is an important aspect of the algorithm. It can be the same constant value for all the ants. Generally, it is required that the amount of pheromone deposited by an ant to be a non-increasing function of the path length [19].

An ACO algorithm iteratively performs a loop containing two basic procedures, one is how the ants construct solutions of the problem which is to be solved, and the other is how to update the pheromone trails [53]. When using ACO algorithms to solve combinatorial optimization problems, it requires a representation of the problem and the definition of the meaning of pheromone trails as well as the heuristic information. Then a specific ACO algorithm is chose and the parameters are needed to tune. The results of many ACO applications to combinatorial optimization problems show that the best performance is achieved when coupling ACO with a local search algorithm [53]. Once the representation and the pheromone as well as heuristic information are defined, the basic operational flow in ACO is as follows:

1) *Initialize ACO parameters;*
2) *Generate random solutions from each ant's random walk;*
3) *Update pheromone intensities;*

4) *Step 2 to 3 are repeated until a termination condition has been satisfied.*

A summary of the results on applying ACO algorithm to Web service concretization is given in Table 4. Besides the listed items, the references in Table 4 differ from each other with respect to the single objective ACO algorithm, which they were based on, such as ant colony system (ACS) adopted in [76], ant system (AS) adopted in [66], [60], [74], and max-min ant system (MMAS) adopted in [65].

The authors of [76] presented a utility function to measure user satisfactory degree. The method was evaluated by comparison with an exhaustive searching using a test scenario with 9 abstract services. The number of concrete service for each abstract service was in the set [10,20,50,100]. The simulation results showed that the execution time of the proposed algorithm was less than that of the exhaustive searching.

In [66], different pheromones were used to denote different QoS attributes. The simulation used 8 abstract services and the number of concrete services corresponding to each abstract service was randomly from 0 to 25. The experimental results showed the dynamic ant colony optimization algorithm had better performance than typical ant colony algorithm.

The study [60] modeled the Web service selection with QoS global optimization problem as a multi-objective optimization problem with user constraints. A multi-objective chaos ACO was proposed to solve it. The chaos variable was used to improve the efficiency. To evaluate the proposed algorithm, the authors compared with a multi-objective GA and a multi-objective ACO algorithm according to three test groups. The number of abstract services and the number of concrete services in each group were (10,5), (20,10), and (20,20). The number of optimal solution and running time were evaluated. The simulation indicated that the proposed algorithm was able to find more optimal solutions and use less time than that of the others.

The authors of [74] used a k-tuple pheromone to represent k objectives. A strategy was used to decompose a composite service with a general flow structure into parallel execution paths.

The authors of [65] proposed a culture max-min ant system algorithm to solve the Web service selection problem. A comprehensive evaluation model based on generic QoS and domain QoS was designed. The generic QoS model was used to evaluate the QoS of composite services, and the domain QoS model was used to conquer the over-constrained problem of Web service composition. A scenario with 10 abstract services and each abstract service has 50 concrete services was used to test the performance of the proposed algorithm. The experimental results showed that the solutions found by the proposed algorithm were much better than solutions searched by an ant colony system algorithm and a max-min ant system algorithm.

5.2 Genetic Algorithm

Genetic algorithms belong to the larger class of evolutionary algorithms, which generate approximate solutions to optimization and search problems by using techniques inspired by the principles of natural evolution: selection, crossover, and mutation. GA is a powerful tool to solve combinatorial optimization problems [54]. It is an iterative procedure based on a constant-size population. In a GA, a population of strings (called

TABLE 4: Summary of results applying ACO to Web service concretization

Articles	Heuristic information used	Pheromone updating was applied to	Key parameters	Limitations and highlights
[76]	C, RT, Av	the best solution	$\alpha = 1$, $\beta = 2$, $\rho = 0.05$	Effects of the number of abstract services are not taken into account. Further investigation is required to deal with all situations of clone ants.
[66]	C, RT, Av, Re	the most satisfying solution	$\alpha = 2$, $\beta = 2$	The setting of parameter is not given and further performance evaluation is required.
[60]	C, RT, Re	the solution in known Pareto Front	$\alpha = 1$, $\beta = 1$, $\rho = 0.3$	There is a need to use different metrics to measure the performance of multi-objective optimization.
[74]	C, RT, Av, Re	all the solutions	$\alpha \in [2, 3, 4, 5]$, $\beta \in [4, 8, 9]$, $\rho = 0.7$	All the abstract execution paths are not consistent with the definition, especially those in the experimental part. The optimal of abstract execution paths cannot guarantee the global QoS constraints of a composite service.
[65]	nine generic QoS attributes including C, RT, RE, Av, and five kind of domain QoS attributes	the top solutions and other remaining solutions using different updating rule	$\alpha = 1$, $\beta = 4$, $\rho = 0.5$	Effects of the changing of the number of abstract services and the number of concrete services are not taken into account. There is a need to devise different scenarios to test scalability.

chromosomes or the genotype of the genome), which are encoded as candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves toward better solutions. Fig. 4 shows how the

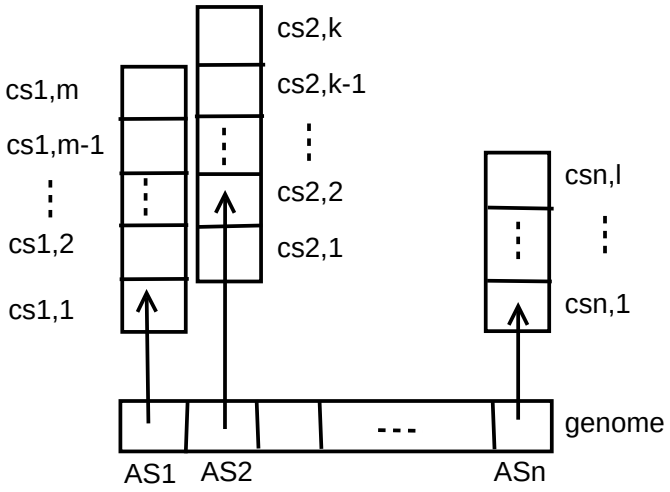

Fig. 4: Genome Encoding

genome is made.

Each genome is associated with a fitness value based on a fitness function that indicates how close it comes to meeting the overall specification, when compared to other genomes in the population. The fitness value of an individual is also an indication of its chances of survival and reproduction in the next generation. The fitness can be measured in a variety of ways: a distance, an error, or a time interval. When using GA to solve Web service concretization problems, the fitness function always corresponds to QoS attributes. An example of fitness function for a genome g is given by (9).

$$F(g) = \frac{w_1 * C(g) + w_2 * RT(g)}{w_3 * Av(g) + w_4 * Re(g)} + w_5 * pf \tag{9}$$

In (9), QoS factors (namely, C(g), Rt(g), Av(g), Re(g)) are normalized in the interval $[0, 1]$ according to (1) and (2). $w_1, \ldots, w_5$ are real, positive weighting factors. In particular, $w_1, \ldots, w_4$ indicate the importance of a QoS attribute which are given by a user, while w_5 weights the penalty factor. The variable pf indicates the penalty factor for individuals that violate constraints.

A typical GA requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain. Once the genetic representation and the fitness function are defined, a GA executes five steps:

1) *Initialization. Traditionally, the population is formed by a group of randomly generated individual solutions.*
2) *Evaluation. The fitness of each individual in the population is evaluated.*
3) *Selection. Individual solutions are selected by the measure of fitness to breed a new generation.*
4) *Evolution. New individuals are created through crossover and mutation operations. The new population is composed of the individuals in the new generation and a few individuals (selected probabilistically) from the previous generation.*
5) *Termination. Steps 2 to 4 are repeated until a termination condition has been reached.*

A summary of the results on applying GA to Web service concretization is given in Table 5.

The study [14] proposed a genetic algorithm with static and dynamic penalty strategy in the fitness function. The experimental results showed that GA was preferred instead of IP when a large number of concrete services were available for each abstract service. When the number of concrete services was small, IP was preferred.

The study [16] proposed a genetic algorithm for cost-driven Web service selection, which only considered cost of concrete services. The test object was a scenario with 20 abstract services, and each abstract service had different number of concrete services. The experimental results showed that the genetic algorithm could work and get better performance when compared with local service selection.

An initial population policy and a mutation policy were proposed to direct the evolution of the genetic algorithms [73]. The performance was evaluated among the genetic algorithms with relation matrix coding scheme and one dimension coding scheme, the genetic algorithms with the initial population policy and population which was randomly created, and the genetic algorithms with the proposed mutation policy and

TABLE 5: Summary of results applying GA to Web service concretization

Articles	Fitness function used	Encoding scheme	Selection operator	Crossover operator	Mutation operator	Limitations and highlights
[14]	C, RT, Re, Av, the current generation and the maximum number of generations	integer array coding	roulette wheel selection and elitism selection	two points crossover, probability was 0.7	randomly selected a gene and randomly replaced with another one, probability was 0.01	The randomly replacement in mutation operator is not efficient.
[16]	C, and a constant	integer array coding	roulette wheel selection and elitism selection	single point crossover, probability was 0.8	The probability of mutation was for the locus of chromosome, mutation probability was 0.05	It is difficult to decide the value of the constant in the fitness function. Further investigation is required to design the fitness function and other QoS attributes should be considered. The experiments was too simple.
[73]	C, RT, Re, Av	relation matrix coding	roulette wheel selection	special crossover operator, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.1	The crossover or mutation operators may generate illegal individuals frequently, and the proposed algorithm does not treat distributed service deployment.
[26]	C, RT, Re, Av, and a penalty item	tree-coding	roulette wheel selection	single point crossover, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.01	Each gene has to record the information of its children and father, which make the length of chromosomes longer and more complicated
[2]	C, RT, Re, Av, and reputation	integer array coding	rank-based selection and elitism selection	single point crossover, probability was 0.9	randomly selected a gene in each chromosome and mutated, probability was 0.08	The repairing operator randomly selects and assigns values, the efficiency is low and it cannot guarantee of finding the feasible solution in the maximal repairing times.
[44]	C, RT, Re, reputation, security, and a penalty item	integer array coding	tournament selection	two points crossover, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.2	The hybrid genetic algorithm is based on exploration of the neighborhood, and the percentage of the neighborhood is a key factor to determine.
[40]	C, RT, Re, Av, and a penalty item	relation matrix coding	roulette wheel selection	special crossover operator, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.1	More scenarios are required to test the proposed algorithm. The investigation is required to escape from frequently created illegal individuals by the crossover and mutation operator.
[37]	C, RT, Re, Av, and reputation	did not provide	did not provide	constrained single point crossover, probability was 0.8	the selection of mutation focused on one specific gene of the entire chromosome, probability was 0.001	The proposed algorithm is required to test more scenarios and further comparisons are needed.
[57]	C, RT, Re, Av, total number of constraint violations in a solution, and the maximal number of possible constraint violations	integer array coding	roulette wheel selection	a knowledge-based one, probability was 0.9	randomly selected a gene in each chromosome and mutated, probability was 0.15	The performance is not stable, further improvement is required to modify the local optimizer.
[59]	C, RT, Re, Av, the reversed index number of qualitative pattern, and some violation value with respect to quantitative constraints	binary string coding	according to a defined probability to select individuals	uniform crossover	randomly chose a single bit of a string to mutate	The structure of execution plan is focus on sequential flow. Large number of service candidates leads to long length of the gnome, and the readability of the chromosomes is fairly weak.

other policies. The experimental results showed that the proposed genetic algorithm could get more excellent composite service plan than standard genetic algorithm.

A novel tree-coding genetic algorithm for QoS-aware service composition was investigated in [26]. The tree-coding could simultaneously express multiple types of composite relationships and could re-plan process at runtime. The performance was evaluated by comparing with one-dimensional coding genetic algorithms. The number of abstract services varied from 9 to 27 with 3 as step length. The experimental results showed that the tree-coding genetic algorithm was effective and faster than one dimensional coding genetic algorithm.

The study [2] presented a repair genetic algorithm to address the Web service composition optimization problem in the presence of domain constraints and inter service dependencies and conflicts. The proposed genetic algorithm used a repair operator to fix up the infeasible individuals such that all the individuals in the population were always feasible. The repair operator was an iterative heuristic approach which randomly selected a gene and randomly chose a value from the set of minimal conflict values for the selected gene. The performance was evaluated by testing the effectiveness and scalability of the proposed repair genetic algorithm, and compared with genetic algorithm without repairing. The experimental results showed that the repair genetic algorithm was scalable and effective.

A tabu search method and a hybrid genetic algorithm were proposed to solve QoS-aware service composition [44]. The hybrid genetic algorithm used a local improvement procedure, which was based on an iterative neighborhood search. The performance of a basic genetic algorithm, an iterative steepest descent methods, a tabu search based algorithm, and a hybrid genetic algorithm was compared with each other. The experimental

results showed that the hybrid genetic algorithm performed better than the basic genetic algorithm for small and medium problem sizes, and that the tabu search based algorithm was not better except for small problem instances and short run times.

The study [40] addressed Web service selection with global QoS constraints using a genetic algorithm with enhanced initial population policy and an evolution policy. An expectation value was used to control the population diversity. The enhanced initial population policy and the evolution policy were used to improve the convergence. The performance was evaluated with regard to the coding scheme, the population diversity handling, the enhanced initial population, and the evolution policy. The standard genetic algorithm was used to compare with the proposed one. The experimental results showed that the proposed genetic algorithm could improve the fitness and promote the convergence.

The authors of [37] proposed a method for incorporating the genetic algorithm and rough set theory, and exemplified by solving the problem of Web services composition. The proposed approach was compared with standard GA and exhaustive enumerations. The experimental results showed that the execution time of the proposed genetic algorithm was not limited to the number of components, and the converged performance and hit rate were much better than that of the other two.

The study [57] modeled the Web service selection problem as a constrained combinatorial optimization problem, and a hybrid genetic algorithm was proposed to solve it. The dependency constraint and conflict constraint among Web services were considered. A local optimizer was used to improve the individuals in the initial population. The performance evaluation was performed by comparing the proposed algorithm with penalty-based and repairing-based genetic algorithms. Three sets of test problems with different number of abstract services, concrete services, and constraints were tested. The experimental results showed the hybrid algorithm was as scalable as the other two genetic algorithms, and could find better quality of solutions. The computation time of the hybrid algorithm did not change as the number of concrete services changed and increased linearly when the number of abstract services or when the number of constraints increased.

In [59], a service composition model considering quantitative and qualitative non-functional properties was presented. The global optimization with local selection algorithm and genetic algorithm were proposed to conduct service composition. The performance was evaluated using a public available dataset and synthetic dataset. The experimental results showed that the quality of GA was better than the global optimization with local selection algorithm in most cases.

5.3 Particle Swarm Optimization

Particle swarm optimization is one of the evolutionary computational techniques. It has been widely used to solve multi-objective optimization problems because it has a small number of parameters, strong robustness, and it is simple and easy to implement. The algorithm finds the optimal solution through iterations after initializing a group of random particles.

The particles update their speed and position through individual best and global best according to (10).

$$\begin{aligned} x_{id}(t+1) &= x_{id}(t) + v_{id}(t+1) \\ v_{id}(t+1) &= \omega v_{id}(t) + c_1\gamma_1(P_{id}(t) - x_{id}(t)) + c_2\gamma_2(P_{gd}(t) - x_{id}(t)) \end{aligned} \tag{10}$$

where ω is the inertia weight that controls the exploration and exploitation of the search space. c_1 (cognition coefficient) and c_2 (social coefficient) are the acceleration constants which change the velocity of a particle towards the P_{id} and P_{gd} [53]. γ_1 and γ_2 are two mutually independent random functions. P_{id} is an individual best, P_{gd} is a global best, x_{id} and v_{id} are the speed and position of d-dimensional particle i, and t means the t-th generation.

The basic operation of PSO is given by,

1) *Initialization. The swarm population is formed.*
2) *Evaluation. The fitness of each individual particle is evaluated.*
3) *Modification. The best position of each particle, the best position of the whole swarm and each particle's velocity are modified.*
4) *Update. Move each particle to a new position.*
5) *Termination. Step 2 to 4 are repeated until a termination condition has been satisfied.*

A summary of the results on applying PSO to Web service concretization is given in Table 6.

The study [24] proposed a multi-objective discrete PSO algorithm for Web selection. The performance of the proposed algorithm was compared with an exhaustive method. The experimental results showed the PSO algorithm could get low computation cost and high quality solutions.

The authors of [50] investigated a discrete multi-objective PSO for optimal Web service composition. The performance was verified by comparing with a multi-objective GA (named NSGAII). The number of abstract

TABLE 6: Summary of results applying PSO to Web service concretization

Articles	Fitness function used	The presentation of particle	Key parameter	Limitations and highlights
[24]	did not provide	N-tuple	ω was computed according to constraints	The experiments cannot show the efficiency of the proposed algorithm, and the fitness function and some parameters are not provided.
[50]	C, RT	N-dimensional vector	$\omega = 0.0275$, $c_1 = 0.3$, $c_2 = 0.49$	Further performance evaluation is required, especially various performance metrics are needed to measure performance of multi-objective optimization algorithms.
[64]	C, RT, Av, and reputation	N-dimensional vector	ω, c_1 and c_2 were self-adaptive	The adaptive termination of algorithms is not considered.
[75]	C, RT, Re, Av	N-dimensional vector	ω, c_1 and c_2 were self-adaptive	Further experiments are required to make sure the results are not biased by the QoS attributes of concrete services.

services was 50, 20 and 5, and the number of concrete services varied from 4 to 50. The experimental results showed that the proposed PSO algorithm used more execution time and created more number of Pareto solutions than that of NSGAII.

The authors of [64] presented an improved PSO algorithm with a non-uniform mutation strategy, an adaptive weight adjustment strategy, and a local best first strategy for QoS-aware Web service selection. These strategies were used to enhance the population diversity and improve convergence speed in global and local level. The performance of the proposed algorithm with each strategy was evaluated, and comparisons with other three algorithms were also presented. The number of abstract services was from 10 to 30 with step of 5, and the number of candidate services for each abstract service was randomly obtained from 10 to 20. The experimental results showed that the proposed strategies could improve the convergence and fitness values to overcome prematurity to some extent.

The study [75] proposed an immune optimization algorithm based on PSO for QoS-driven Web service composition with global constraints. An improved local best first strategy and a perturbing global best strategy were also presented. The performance of the proposed algorithm was evaluated by comparing with the standard PSO and the GA [40]. The task numbers are 20, 40, 60, and 80 for composite Web service instances and each task contains 15 candidate services. The mean best fitness and standard deviation of the optimal fitness values were tested. The experimental results showed that the proposed algorithm was better than the other two in terms of search ability, convergence property, and stability.

5.4 Combination of Different Bio-inspired Algorithms

A summary of the results on applying the combination of bio-inspired algorithms to Web service concretization is given in Table 7.

In [38], a hybrid genetic particle swarm algorithm was proposed to solve Web service composition problem. The algorithm was designed to balance the global optimization iteration numbers and local optimization iteration numbers. The performance was verified by comparing with a balanced multi-objective GA. The experimental results indicated that the hybrid algorithm improved the local result optimization.

TABLE 7: Summary of results applying combination of bio-inspired algorithm to Web service concretization

Articles	Fitness function used	Reason(s) for combining	Limitations and highlights
[38]	did not provide details	The GA was used to search throughout the problem space, and the PSO was used to enhance local search ability.	There is a need to show more details of the experiments, and find most appropriate and robust parameters for testing.
[67]	C, RT	The GA was used to set the key parameters of ACO	The experiment cannot indicate the performance of the algorithms. The QoS dataset is not provided. There is a need to design more scenarios for performance testing.
[23]	did not provide details	The swarm in SPSO was often not convergent to the global optimal position but a local best position, and SA was convergent to the optimum but its convergent speed was slow.	The parameters of the proposed algorithm and the QoS dataset are not provided. Further investigations are required to test the performance of the proposed algorithms in terms of the changing number of abstract services and concrete services.

A combination of ACO and GA for Web service composition was presented in [67]. The performance of the proposed algorithm was evaluated by comparing with a GA and an immune algorithm. The test object was a composite service with 10 tasks and each task with 35 candidate services. The experimental results showed that the combination algorithm had better performance than the other two algorithms in terms of optimal solution and iteration times.

A cooperative evolution algorithm that consisted of stochastic particle swarm optimization (SPSO) and simulated annealing (SA) was presented to solve the Web service selection problem [23]. Furthermore, a multi-objective SPSO for Web service composition with global constraints was proposed. The SPSO was used to search the local optimum rapidly, while SA was adopted to make the individual jump off the local optimum with greater probability. The performance was evaluated by comparing the performance of SPSO, SA, and cooperative algorithm according to the quality of solution, computation complexity, and convergence rate. The experimental results indicated that cooperative algorithm was better than the other two.

5.5 Discussions

A variety of studies suggest that parameter selection has a strong impact on the performance of a bio-inspired algorithm. They argue that inept setting of parameters of bio-inspired algorithms may result in bad performance of service composition, but only a few of the existing studies [64], [65], [74], [75] gave an explicit investigation about how the setting of parameters affected the performances of service composition based on bio-inspired algorithms. For example, the design of GA has a great influence on its behavior and performance [51]. It is necessary for GA to accord with the characteristic of the service composition in order to get better performance. By reviewing a number of studies on bio-inspired algorithm-based Web service composition, we summarize the main control parameters involved in each algorithm as listed in Table 8, which can be referred when using bio-inspired algorithms to deal with service composition problems.

The ant colony algorithm focuses on theoretical problems such as parameter initialization and information updating. It can be run continuously and adapts to changes in real time. It may have some advantages over PSO and GA approaches if the graph changes dynamically. On the other hand, the efficiency of the ant colony algorithm is very low. In genetic algorithms, the mutation and crossover operator are relatively fixed, and random search without guide will cause degeneration. ACO and PSO share many common points with GA, such as a randomly generated population, and a fitness function to evaluate and search. Unlike GA, however, ACO and PSO have no evolution operators such as crossover or mutation. In addition, PSO is

TABLE 8: Main control parameters in three bio-inspired algorithms

Algorithm	Main control parameters
ACO	number of ants
	number of iterations
	influence of heuristic (β)
	influence of pheromone (α)
	pheromone evaporation coefficient (ρ)
	definition of heuristic factor (η)
	pheromone updating strategy
GA	population size
	number of generations
	encoding scheme
	selection operator
	crossover operator
	crossover probability
	mutation operator
	mutation probability
	definition of fitness function
PSO	population size
	inertia weight (ω)
	number of generations
	particle initial position
	particle initial velocity
	acceleration factor (c_1 and c_2)
	definition of fitness function

easy to implement and there are a few more parameters to adjust. All these bio-inspired algorithms have the problem of premature convergence, and they will be easily trapped into the local optimality. So many efforts are made to improve the performance through combining them.

Normally, the Web service composition needs to be performed and completed in a short time period and the quality of solutions needs to fully satisfy a user's requirements. Many studies into Web service composition based on bio-inspired algorithms are attempting to reduce processing time for a near optimum solution, to improve the quality of solutions, and particularly to avoid being trapped into local optimization. Some studies have carried out comparisons among different types of a kind of algorithm, such as comparing the performance of ACO and chaos ACO [60], comparing the performance of improved GA and standard GA [2], [26], [37], [40], [44], [57], [73], and others have compared different types of algorithms, such as comparing the performance of combination of ACO and GA with that of standard GA [67].

6 Experiments and Analysis

We have conducted extensive simulations to evaluate the performance of ACO and GA, which we will describe in this section. We have done some work to apply ACO and GA for Web service composition [62], [63], however, we did not compare the two algorithms. In this paper, the comparisons between ACO and GA are conducted.

The ACO algorithm was based on ant colony system. Both the number of ants in ACO and the size of population in GA are 20. In this experiment, trial testing method is adopted to determine most suitable values for all parameters in ACO and GA, considering other researchers' earlier experiments. We define stop criteria for both algorithms: iterate until a maximum number of iteration (1000 in our case study) is reached, alternatively, iterate until the best fitness value remains unchanged for a given number of iteration (50 in our case study).

6.1 The Dataset

In our evaluation, we experimented with two types of datasets as used in [5], [59]: the real-word Web service QoS dataset (QWS) and synthetic datasets (RQoS). The publicly available dataset QWS [3] comprises measurements of 9 QoS attributes for 2507 real-world Web services. It is utilized as the QoS dataset of candidate services and specifically we use response time, availability, throughput, successability, and reliability as the QoS attributes. More details about this dataset can refer to [4]. For synthetic datasets, they are randomly generated from the interval $[1, 100]$. Except for response time and throughput, others' units are percent.

6.2 Evaluation Methodology

For the purpose of our evaluation, we considered some scenarios, where a composite service comprises n abstract services (n varies in our experiments between 10 and 50, in increments of 10). There are m concrete services in each service candidate set (m varies in our experiments between 100 and 1000, in increments of 100). Thus, we randomly partitioned each of the aforementioned datasets into n service candidate sets. The objective is to maximize the overall score of execution path which is given in (5). We then recorded the required computation time (the average of 50 executions) by ACO and GA. As ACO and GA are sub-optimal method, we have evaluated the quality of the results obtained by these two bio-inspired algorithms through comparing it with the optimal results obtained by the MIP approach. We compute the optimality of the results of ACO and GA by comparing the overall fitness value (U_{bio}) of the selected services to the overall fitness value (U_{global}) of the optimal selection obtained by the MIP approach, that is $optimality = U_{bio}/U_{global}$. We used the open source integer programming system *lpsolve* version 5.5 [12].

6.3 Performance Comparison

In the following we present a performance comparison of ACO and GA with respect of the number of candidate services, and the number of abstract services. In Fig. 5, we compare the performance of ACO and GA with respect to the number of candidate services for each abstract service. The number of service candidates per abstract service varies from 100 to 1000, while the number of abstract service is set to 10. The results indicate that the required computation time of GA increases very slowly, this makes GA more scalable than ACO.

In the experiment shown in Fig. 6, we study the performance of both algorithms with respect to the number of abstract services in the composition. Again, we observe that GA outperforms ACO in all datasets. The reason is that each ant needs to compute the probability of next node when it finds the path and such computation costs much time.

Next, we present the evaluation of the quality of the results obtained by ACO and GA in terms of the achieved optimality. Table 9 shows the optimality achieved with different datasets and a varying number of

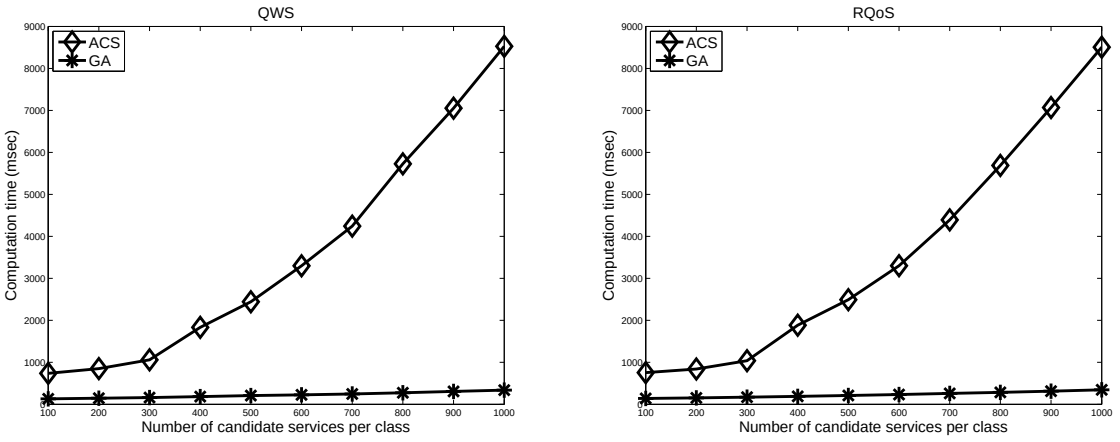

Fig. 5: Performance vs. number of candidate services

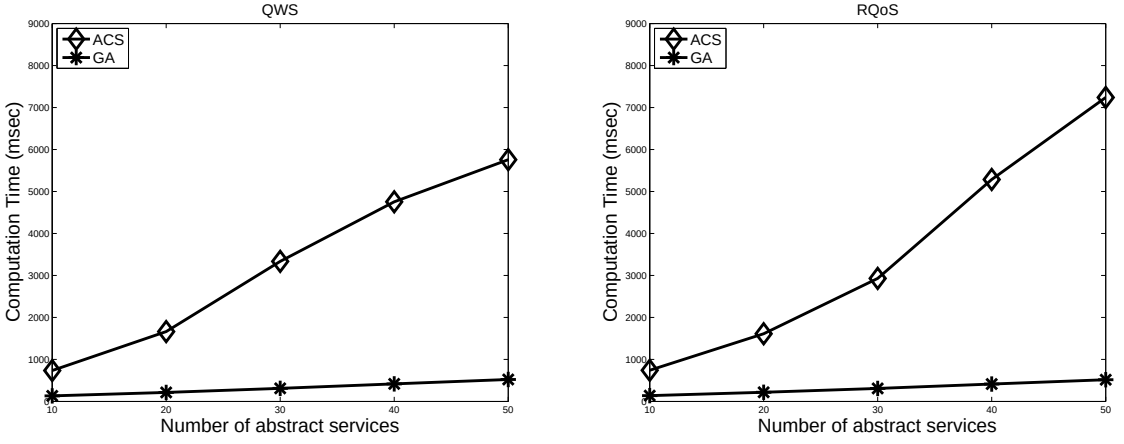

Fig. 6: Performance vs. number of abstract services

TABLE 9: Mean of optimality (%)

Scenarios	QWS		RQoS	
	ACS	GA	ACS	GA
n is fixed at 10, m varies between 100 and 1000, in increments of 100	100	100	100	100
	99.9819	100	100	100
	100	100	100	100
	100	100	100	100
	100	100	100	100
	99.9926	100	100	100
	100	100	99.9836	100
	99.9936	100	100	100
	99.9785	100	100	100
	100	100	100	100
m is fixed at 50, n varies between 10 and 50, in increments of 10	100	100	100	100
	99.9800	100	99.9525	100
	99.8678	100	99.7790	100
	99.7149	100	98.0089	100
	99.5823	100	100	100

candidate services and a varying number of abstract services. The results indicate that GA and ACO were able to achieve very close-to-optimal results.

Table 10 gives the mean of FRIT with different datasets and a varying number of candidate services and

TABLE 10: Mean of FRIT

Scenarios	QWS		RQoS	
	ACS	GA	ACS	GA
n is fixed at 10, m varies between 100 and 1000, in increments of 100	1	6	1	7
	2	7	1	7
	1	7	1	6
	1	6	1	7
	1	6	1	7
	2	6	1	7
	1	6	2	7
	2	7	1	7
	1	7	1	7
	1	7	1	6
m is fixed at 50, n varies between 10 and 50, in increments of 10	1	6	1	6
	4	9	2	9
	11	11	10	11
	16	13	16	13
	20	14	31	14

a varying number of abstract services. The value of 'FRIT' is iteration numbers when the best fitness value appeared and from this iteration the value of the best fitness will not change. The results show that both algorithms need more iteration numbers to get the best fitness value when the number of abstract services increases. When the number of concrete services increases, the values of FRIT of both algorithms almost not change and ACS can get the best fitness value almost in the first iteration.

7 Conclusion

In this paper, the Web service concretization based on bio-inspired algorithms is discussed. The existing QoS-based Web service concretization approaches can be categorized into two groups: local optimization and global optimization. The local optimization cannot consider global constraints, or it can be achieved by using decomposition methods. The global optimization approaches construct composite services from a global level to satisfy the QoS constraints. Several optimal and sub-optimal algorithms have been proposed to solve QoS-based Web service concretization problems. Particularly, meta-heuristics-based algorithms intend to find a near-to-optimal solution. These algorithms have good scalability and can decrease the computational complexity in selecting services if compared with the optimal algorithms.

The applications of ant colony algorithm, genetic algorithm, and particle swarm optimization algorithm to Web service concretization have been described in details. Comparisons between the former two algorithms have been conducted. While some progress has been made in recent years in the area of bio-inspired algorithms-based Web service concretization, there remains much potential for further efforts to be made in this area.

One future challenge is to establish a standard framework for designing and testing bio-inspired algorithm-based service concretization. Currently, there is no such framework available. The existing studies presenting comparisons among bio-inspired algorithms had limited themselves to their own simulated environment and test data set. Their experimental results represented only a small number of specific scenarios. We cannot tell which algorithm performs better in a certain scenario. Guidelines are also needed for service developers to choose the optimal algorithm to offer composite services, which could enable the service users to get better performance based on their own requirements and preferences.

A second major challenge is to explore key features and mechanisms of biological systems to be applied in cloud computing and big data areas. Due to the deluge of enormous sources of data and the development of cloud computing, applications based on data-intensive services have become the most challenging type of applications in service-oriented computing. Data access, data analysis, and data manipulation may require a nontrivial composition of a series of data retrieval and process executions. The service-based strategy provides maximal flexibility when designing data-intensive applications. These data services are different from traditional services because they handle huge amounts of data, so their QoS attributes might be quite different. Because these data-intensive services are normally larger scale and usually more complex, we deem that bio-inspired mechanisms can also facilitate data-intensive service provision with convincing outcomes. The design of the three bio-inspired algorithms for data-intensive service provision is currently under way.

ACKNOWLEDGMENT

The authors gratefully acknowledge the help of Dr. Madeleine Strong Cincotta in the final language editing of this paper.

REFERENCES

[1] R. Aggarwal, K. Verma, J. Miller, and W. Milnor, "Constraint driven web service composition in meteor-s," in *Proceedings of IEEE International Conference on Services Computing (SCC '04)*, 2004, pp. 23–30.
[2] L. Ai and M. Tang, "Qos-based web service composition accommodating inter-service dependencies using minimal-conflict hill-climbing repair genetic algorithm," in *IEEE Fourth International Conference on eScience (eScience '08)*, 2008, pp. 119–126.
[3] E. Al-Masri and Q. H. Mahmooud, "The qws dataset," 2008, http://www.uoguelph.ca/~qmahmoud/qws/index.html.
[4] E. Al-Masri and Q. H. Mahmoud, "Investigating web services on the world wide web," in *Proceedings of the 17th international conference on World Wide Web (WWW '08)*. New York, NY, USA: ACM, 2008, pp. 795–804.
[5] M. Alrifai, T. Risse, and W. Nejdl, "A hybrid approach for efficient web service composition with end-to-end qos constraints," *ACM Transactions on the Web*, vol. 6, no. 2, pp. 7:1–7:31, 2012.
[6] D. Ardagna and B. Pernici, "Global and local qos guarantee in web service selection," in *Business Process Management Workshops*, ser. Lecture Notes in Computer Science, C. Bussler and A. Haller, Eds. Berlin, Heidelberg: Springer-Verlag, 2006, vol. 3812, pp. 32–46.
[7] ——, "Adaptive service composition in flexible processes," *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 369–384, May 2007.
[8] D. Bacon, S. Graham, and O. Sharp, "Compiler transformations for high-performance computing," *ACM Computing Surveys*, vol. 26, no. 4, pp. 345–420, December 1994.
[9] S. Balasubramaniam, D. Botvich, R. Carroll, J. Mineraud, T. Nakano, T. Suda, and W. Donnelly, "Biologically inspired future service environment," *Computer Networks*, vol. 55, no. 15, pp. 3423–3440, 2011.
[10] B. Benatallah, M. Dumas, Q. Sheng, and A. Ngu, "Declarative composition and peer-to-peer provisioning of dynamic web services," in *Proceedings of 18th International Conference on Data Engineering*, 2002, pp. 297–308.
[11] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz, "Heuristics for qos-aware web service composition," in *International Conference on Web Services (ICWS '06)*, 2006, pp. 72–82.
[12] M. Berkelaar, K. Eikland, and P. Notebaert, "lpsolve: Open source (mixed-integer) linear programming system," 2004, http://lpsolve.sourceforge.net/.
[13] P. Bless, D. Klabjan, and S. Chang, "Heuristics for automated knowledge source integration and service composition," *Computers & Operations Research*, vol. 35, no. 4, April 2008.
[14] G. Canfora, M. Penta, R. Espositio, and M. Villani, "An approach for qos-aware service composition based on genetic algorithms," in *Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO '05)*. New York, NY, USA: ACM, 2005, pp. 1069–1075.
[15] G. Canfora, M. Penta, R. Esposito, and M. Villani, "A lightweight approach for qos aware service composition," in *Proceedings of 2nd International Conference on Service Oriented Computing (ICSOC '04)*. New York, NY, USA: ACM, 2004, pp. 36–47.
[16] L. Cao, M. Li, and J. Cao, "Cost-driven web service selection using genetic algorithm," in *Internet and Network Economics*, ser. Lecture Notes in Computer Science, X. Deng and Y. Ye, Eds. Berlin, Heidelberg: Springer-Verlag, 2005, vol. 3828, pp. 906–915.
[17] V. Cardellini, E. Casalicchio, V. Grassi, and F. Presti, "Efficient provisioning of service level agreements for service oriented applications," in *2nd International Workshop on Service Oriented Software Engineering: in Conjunction with the 6th ESEC/FSE Joint Meeting (IW-SOSWE '07)*. New York, NY, USA: ACM, 2007, pp. 29–35.
[18] D. Comes, H. Baraki, R. Reichle, M. Zapf, and K. Geihs, "Heuristic approaches for qos-based service selection," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science, P. Maglio, M. Weske, J. Yang, and M. Fantinato, Eds. Berlin, Heidelberg: Springer-Verlag, 2010, vol. 6470, pp. 441–455.
[19] M. Dorigo and T. Stutzle, *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press, 2004.
[20] S. Dustdar and W. Schreiner, "A survey on web services composition," *International Journal of Web and Grid Services*, vol. 1, no. 1, pp. 1–30, Aug. 2005.
[21] T. Erl, *Service-Oriented Architecture: Concepts, Technology, and Design*. United States: Prentice Hall/PearsonPTR Heg USA, 2005.
[22] I. Estevez-Ayres, P. Basanta-Val, M. Garcia-Valls, J. Fisteus, and L. Almeida, "Qos-aware real-time composition algorithms for service-based applications," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 3, pp. 278–288, August 2009.
[23] X. Fan, X. Fang, and C. Jiang, "Research on web service selection based on cooperative evolution," *Expert Systems with Applications*, vol. 38, no. 8, pp. 9736–9743, August 2011.
[24] X. Fan, C. Jiang, and X. Fang, "An efficient approach to web service selection," in *Web Information Systems and Mining*, ser. Lecture Notes in Computer Science, W. Liu, X. Luo, F. Wang, and J. Lei, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, vol. 5854, pp. 271–280.
[25] A. Ferreira, K. Kritikos, and B. Pernici, "Energy-aware design of service-based applications," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science, L. Baresi, C. Chi, and J. Suzuki, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, vol. 5900, pp. 99–114.
[26] C. Gao, M. Cai, and H. Chen, "Qos-aware service composition based on tree-coded genetic algorithm," in *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, 2007, pp. 361–367.
[27] A. Huang, C. Lan, and S. Yang, "An optimal qos-based web service selection scheme," *Information Sciences*, vol. 179, no. 19, pp. 3309–3322, September 2009.
[28] D. Ivanović, M. Carro, and M. Hermenegildo, "A constraint-based approach to quality assurance in service choreographies," in *Proceedings of 10th International Conference on Service-Oriented Computing (ICSOC '12)*, 2012, pp. 252–267.
[29] M. Jaeger, "Optimising quality-of-service for the composition of electronic services," Ph.D. dissertation, Berlin University of Technology, January 2007.
[30] M. Jaeger and G. Muehl, "Qos-based selection of services: The implementation of a genetic algorithm," in *Proceedings of KiVS 2007 Workshop: Service-Oriented Architectures and Service-Oriented Computing (SOA/SOC)*, T. Braun, G. Carle, and B. Stiller, Eds. Bern, Switzerland: VDE, 2007, pp. 359–370.
[31] M. Jaeger, G. Muehl, and S. Golze, "Qos-aware composition of web services: An evaluation of selection algorithms," in *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds. Berlin, Heidelberg: Springer-Verlag, 2005, vol. 3760, pp. 646–661.
[32] B. Kitchenham, "Guidelines for performing systematic literature reviews in software engineering," Tech. Rep. EBSE-2007-001, July 2007, http://www.dur.ac.uk/ebse/resources/.

[33] Z. Kobti and Z. Wang, "An adaptive approach for qos-aware web service composition using cultural algorithms," in *AI 2007: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, M. Orgun and J. Thornton, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, vol. 4830, pp. 140–149.

[34] J. Lee, "Matching algorithms for composing business process solutions with web services," in *E-Commerce and Web Technologies*, ser. Lecture Notes in Computer Science, K. Bauknecht, A. Tjoa, and G. Quirchmayr, Eds. Berlin, Heidelberg: Springer-Verlag, 2003, vol. 2738, pp. 393–402.

[35] F. Li, F. Yang, K. Shuang, and S. Su, "Q-peer: A decentralized qos registry architecture for web services," in *Service-Oriented Computing - ICSOC 2007*, ser. Lecture Notes in Computer Science, B. Krămer, K. Lin, and P. Narasimhan, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, vol. 4749, pp. 145–156.

[36] L. Li, P. Yang, L. Ou, Z. Zhang, and P. Cheng, "Genetic algorithm-based multi-objective optimization for qos-aware web services composition," in *Knowledge Science, Engineering and Management*, ser. Lecture Notes in Computer Science, Y. Bi and M. Williams, Eds. Berlin, Heidelberg: Springer-Verlag, 2010, vol. 6291, pp. 549–554.

[37] W. Liang and C. Huang, "The generic genetic algorithm incorporates with rough set theory - an application of the web services composition," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5549–5556, April 2009.

[38] J. Liu, J. Li, K. Liu, and W. Wei, "A hybrid genetic and particle swarm algorithm for service composition," in *Sixth International Conference on Advanced Language Processing and Web Information Technology*, 2007, pp. 564–567.

[39] S. Liu, Y. Liu, N. Jing, G. Tang, and Y. Tang, "A dynamic web service selection strategy with qos global optimization based on multi-objective genetic algorithm," in *Grid and Cooperative Computing - GCC 2005*, ser. Lecture Notes in Computer Science, H. Zhuge and G. Fox, Eds. Berlin, Heidelberg: Springer-Verlag, 2005, vol. 3795, pp. 84–89.

[40] Y. Ma and C. Zhang, "Quick convergence of genetic algorithm for qos-driven web service selection," *Computer Networks*, vol. 52, no. 5, pp. 1093–1104, 2008.

[41] N. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny, "Qos-aware service composition in dynamic service oriented environments," in *Middleware 2009*, ser. Lecture Notes in Computer Science, J. Bacon and B. Cooper, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, vol. 5896, pp. 123–142.

[42] F. Mardukhi, N. NematBakhsh, K. Zamanifar, and A. Barati, "Qos decomposition for service composition using genetic algorithm," *Applied Soft Computing*, vol. 13, no. 7, pp. 3409–3421, February 2013.

[43] T. Nakano, "Biologically inspired network systems: A review and future prospects," *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and reviews*, vol. 41, no. 5, pp. 630–643, September 2011.

[44] J. Parejo, P. Fernandez, and A. Cortes, "Qos-aware services composition using tabu search and hybrid genetic algorithms," *Actas de Talleres de Ingeniera Del Software y Bases de Datos*, vol. 2, no. 1, pp. 55–66, 2008.

[45] C. Patel, K. Supekar, and Y. Lee, "A qos oriented framework for adaptive management of web service based workflows," in *Database and Expert Systems Applications*, ser. Lecture Notes in Computer Science, V. Mařík, W. Retschitzegger, and O. Štěpánková, Eds. Berlin, Heidelberg: Springer-Verlag, 2003, vol. 2736, pp. 826–835.

[46] B. Pernici, S. Siadat, S. Benbernou, and M. Ouziri, "A penalty-based approach for qos dissatisfaction using fuzzy rules," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science, G. Kappel, Z. Maamar, and H. Motahari-Nezhad, Eds., 2011, vol. 7084, pp. 574–581.

[47] C. Pop, V. Chifu, I. Salomie, M. Dinsoreanu, T. David, and V. Acretoaie, "Ant-inspired technique for automatic web service composition and selection," in *12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2010, pp. 449–455.

[48] Y. Qu, C. Lin, Y. Wang, and Z. Shan, "Qos-aware composite service selection in grids," in *Proceedings of 5th International Conference on Grid and Cooperative Computing (GCC '06)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 458–465.

[49] J. Rao and X. Su, "A survey of automated web service composition methods," in *Semantic Web Services and Web Process Composition*, ser. Lecture Notes in Computer Science, J. Cardoso and A. Sheth, Eds. Berlin, Heidelberg: Springer-Verlag, 2005, vol. 3387, pp. 43–54.

[50] H. Rezaie, N. NematBaksh, and F. Mardukhi, "A multi-objective particle swarm optimization for web service composition," in *Networked Digital Technologies*, ser. Communications in Computer and Information Science, F. Zavoral, J. Yaghob, P. Pichappan, and E. El-Qawasmeh, Eds. Springer Berlin Heidelberg, 2010, vol. 88, pp. 112–122.

[51] I. Rojas, J. Gonzalez, H. Pomares, J. J. Merelo, P. A. Castillo, and G. Romero, "Statistical analysis of the main parameters involved in the design of a genetic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics-Part. C: Applications and Reviews*, vol. 32, no. 1, pp. 31–37, February 2002.

[52] A. Ruiz-Cortés, O. Martín-Díaz, A. Durán, and M. Toro, "Improving the automatic procurement of web services using constraint programming," *International Journal of Cooperative Information Systems*, vol. 14, no. 4, pp. 439–467, December 2005.

[53] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2007.

[54] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[55] S. Sun and J. Zhao, "A decomposition-based approach for service composition with global qos guarantees," *Information Sciences*, vol. 199, pp. 138–153, March 2012.

[56] Y. Syu, S. Ma, J. Kuo, and Y. FanJiang, "A survey on automated service composition methods and related techniques," in *IEEE Ninth International Conference on Services Computing (SCC)*, 2012, pp. 290–297.

[57] M. Tang and L. Ai, "A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition," in *Proceeding of the 2010 World Congress on Computational Intelligence*. Washington, DC, USA: IEEE Computer Society, 2010, pp. 268–275.

[58] W. van der Aalst and A. ter Hofstede, "Yawl: Yet another workflow language," *Information Systems*, vol. 30, no. 4, pp. 245–275, June 2005.

[59] H. Wang, P. Ma, and X. Zhou, "A quantitative and qualitative approach for nfp-aware web service composition," in *IEEE Ninth International Conference on Services Computing (SCC)*, 2012, pp. 202–209.

[60] L. Wang and Y. He, "A web service composition algorithm based on global qos optimizing with mocaco," in *Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE 2011)*, ser. Advances in Intelligent and Soft Computing, L. Jiang, Ed. Berlin, Heidelberg: Springer-Verlag, 2010, vol. 111, pp. 79–86.

[61] L. Wang and J. Shen, "Towards bio-inpsired cost minimisation for data-intensive service provision," in *IEEE International Conference on Services Economics*, 2012, pp. 16–23.

[62] ——, "Economical data-intensive service provision supported with a modified genetic algorithm," in *IEEE 2nd International Congress on Big Data*. Washington, DC, USA: IEEE Computer Society, 2013, pp. 358–365.

[63] L. Wang, J. Shen, and G. Beydoun, "Enhanced ant colony algorithm for cost-aware data-intensive service provision," in *IEEE Ninth World Congress on Services*. Washington, DC, USA: IEEE Computer Society, 2013, pp. 227–234.

[64] W. Wang, Q. Sun, X. Zhao, and F. Yang, "An improved particle swarm optimization algorithm for qos-aware web service selection in service oriented communication," *International Journal of Computational Intelligence Systems*, vol. 3, no. 01, pp. 18–30, March 2012.

[65] Z. Wang, Z. Liu, X. Zhou, and Y. Lou, "An approach for composite web service selection based on dgqos," *The International Journal of Advanced Manufacturing Technology*, vol. 56, no. 9-12, pp. 1167–1179, October 2011.
[66] Y. Xia, J. Chen, and X. Meng, "On the dynamic ant colony algorithm optimization based on multi-pheromones," in *Seventh IEEEACIS International Conference on Computer and Information Science (ICIS '08)*, 2008, pp. 630–635.
[67] Z. Yang, C. Shang, Q. Liu, and C. Zhao, "A dynamic web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm," *Computational Information Systems*, vol. 6, no. 8, pp. 2617–2622, August 2010.
[68] G. Ying, A. Ghose, and L. Zheng, "Using constraint hierarchies to support qos-guided service composition," in *International Conference on Web Services (ICWS '06)*, 2006, pp. 743–752.
[69] K. Yoon and H. Land, *Multiple Attribute Decision Making: An Introduction*, ser. Quantitative Applications in the Social Sciences. CA, Thousand Oaks: SAGE Publications, Inc., 1995, vol. 104.
[70] T. Yu, Y. Zhang, and K. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Transactions on the Web*, vol. 1, no. 1, pp. 1–26, May 2007.
[71] M. Zemni, S. Benbernou, and M. Carro, "A soft constraint-based approach to qos-aware service selection," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science, P. Maglio, M. Weske, J. Yang, and M. Fantinato, Eds. Berlin, Heidelberg: Springer-Verlag, 2010, vol. 6470, pp. 596–602.
[72] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311–327, May 2004.
[73] C. Zhang, S. Su, and J. Chen, "A novel genetic algorithm for qos-aware web services selection," in *Data Engineering Issues in E-Commerce and Services*, ser. Lecture Notes in Computer Science, J. Lee, J. Shim, S. Lee, C. Bussler, and S. Shim, Eds. Berlin, Heidelberg: Springer-Verlag, 2006, vol. 4055, pp. 224–235.
[74] W. Zhang, C. Chang, T. Feng, and H. Jiang, "Qos-based dynamic web service composition with ant colony optimization," in *Proceedings of the 34th Annual IEEE International Computer Software and Applications Conference (COMPSAC '10)*. Washington, DC, USA: IEEE Computer Society, 2010, pp. 493–502.
[75] X. Zhao, B. Song, P. Huang, Z. Wen, J. Weng, and Y. Fan, "An improved discrete immune optimization algorithm based on pso for qos-driven web service composition," *Applied Soft Computing*, vol. 12, no. 8, pp. 2208–2216, August 2012.
[76] X. Zheng, J. Luo, and A. Song, "Ant colony system based algorithm for qos-aware web service selection," in *Proceedings of 4th International Conference on Grid Service Engineering and Management (GSEM)*, 2007, pp. 39–50.